Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 23, 2019

B. Lengyel
Ericsson
B. Claise
Cisco Systems, Inc.
July 22, 2018

# YANG Instance Data Files and their use for Documenting Server Capabilities
## draft-lengyel-netmod-yang-instance-data-03

Abstract

This document specifies a standard file format for YANG instance data, that is data that could be stored in a datastore and whose syntax and semantics is defined by YANG models. Instance data files can be used to provide information that is defined in design time. There is a need to document Server capabilities (which are often specified in design time). Defining server capabilities is foreseen as the most important use of YANG instance data files.

Table of Contents



## 1. Terminology

Design time: A time during which a YANG model and the implementation behind it is created. Sometimes in other documents this period is divided into design and implementation time.

Instance Data Set: A named set of data items that can be used as instance data in a YANG data tree.

Instance Data File: A file containing an instance data set formatted according to the rules described in this document.

Target YANG Module: A YANG module for which the instance data set contains instance data, like ietf-yang-library in the examples.

## 2. Introduction

A YANG server has a number of server-capabilities that can be retrieved from the server using protocols like NETCONF or RESTCONF. YANG server capabilities include

- data defined in ietf-yang-library: YANG modules, submodules, features, deviations, schema-mounts ([I-D.ietf-netconf-rfc7895bis])

- datastores supported

- alarms supported ([I-D.ietf-ccamp-alarm-module])

- data nodes, subtrees that support or do not support on-change notifications ([I-D.ietf-netconf-yang-push])

- netconf-capabilities

While it is good practice to allow a client to query these capabilites from the live YANG server, that is often not enough. Most server-capabilities are relatively stable but the fact is that some might change. Looking at the change frequency, we have roughly three categories:

1. only at upgrade, e.g. introduced with a new SW package

2. rarely e.g. due to licensing or HW inserted

3. more frequently e.g. a capability might be dependent on the CPU or traffic load, although that would be most unusual

Most capabilities belong to type 1), some to type 2) and a relatively small set to type 3). Many network nodes only have type 1) or type 1+2) capabilities. Stable capabilities are usually defined by a vendor in design time, before the product is released. While these capabilities can be retrieved from the live server in run-time, there is a strong need to provide the same data already during design time. (Often only a part of all the server capabilities can be made available.)

Often when a network node is released an associated NMS (network management system) is also released with it. The NMS depends on the capabilities of the YANG server. During NMS implementation information about server capabilities is needed. If the information is not available early in some off-line document, but only as instance data from the live network node, the NMS implementation will be delayed, because it has to wait for the network node to be ready. Also assuming that all NMS implementors will have a correctly configured network node available to retrieve data from, is a very expensive proposition. (An NMS may handle dozens of node types.)

Beside NMS implementors, system integrators and many others also need the same information early. Examples could be model driven testing, generating documentation, etc.

As capabilities are often already known in design time and are relativaly stable, it feasible and advantageous to define/document them early. This document specifies a file format for YANG instance data that may be used to provide server capability information, allowing vendors to specify capabilities early, in design time.

The same instance data file format can be used for other purposes, like providing initial data for any YANG module. E.g. a basic set of access control groups can be provided either by a device vendor or an operator using the network device.

## 2.1. Data Life cycle

Data defined or documented in YANG Instance Data Sets may be used for preloading a YANG server with this data, but the server may populate the data without using the actual file in which case the Instance Data File is only used as documentation.

While such data will usually not change, data documented by Instance Data sets MAY be changed by the YANG server itself or by management operations. It is out of scope for this document to specify a method to prevent this. Whether such data changes and if so, when and how, SHOULD be described either in the instance data file description statement or in some other implementation specific manner.

YANG Instance data is a snap-shot of information at a specific point of time. If the data changes afterwards this is not represented in the instance data set anymore, the valid values can be retrieved in run-time via Netconf/Restconf

Notifications about the change of data documented by Instance Data Sets may be supplied by e.g. the Yang-Push mechanism, but it is out of scope for this document.

## 2.2. Delivery of Instance Data

Instance data files SHOULD be available without the need for and before the instalation of a live YANG server e.g. via download from the vendor's website. or any other way together with other product documentation.

### 2.3. Use Case 1: Early Documentation of Server Capabilites

An operator wants to integrate his own, in-house built management system with the network node from ACME Systems. The management integration must be ready by the time the first AcmeRouter is installed in the network. To do the integration the operator needs the list of supported YANG modules and features. While this list could be read from the ietf-yang-library via Netconf, in order to allow time for developing the management integration, the operator demands this information early. The operator will value that this information is available in a standard format, that is actually the same format that can be read later from the node via Netconf.

YANG instance data files are used to provide design time information about server capabilities.

### 2.4. Use Case 2: Preloading Data

There are parts of the configuration that must be fully configurable by the operator, however for which often a semi-standard default configuration will be sufficient.

One example is access control groups/roles and related rules. While a sophisticated operator may define dozens of different groups often a basic (read-only operator, read-write system administrator, security-administrator) triplet will be enough. Vendors will often provide such default configuration data to make device configuration easier for an operator.

Defining Access control data is a complex task. To help the device vendor pre-defines a set of default groups (/nacm:nacm/groups) and rules for these groups to access specific parts of common models (/nacm:nacm/rule-list/rule).

YANG instance data files are used to document and/or preload the default configurationp.

### 2.5. Use Case 3: Dcoumenting Factory Default Settings

Nearly every YANG server has a factory default configuration. If the system is really badly misconfigured or if the current configuration is to be abandoned the system can be reset to this default.

In Netconf the <delete-config> operation can be used to do this, while in Restconf there are plans to introduce a custom operation for this purpose.

The operator curerntly has no way to know what the default configuration actually contains. YANG Instance data can be used to document the factory default configuration.

## 3. Instance Data File Format

Two standard formats to represent YANG Instance Data are specified based on the XML and JSON encoding. The XML format is based on [RFC7950] while the JSON format is based on [RFC7951]. Later as other YANG encodings (e.g. CBOR) are defined further Instance Data formats may be specified.

For both formats data is placed in a top level auxiliary container named "instance-data-set". The purpose of the container, which is not part of the real data itself, is to carry meta-data for the complete instance-data-set.

The XML format SHALL follow the format returned for a NETCONF GET operation. The <data> anydata (which is not part of the real data itself) SHALL contain all data that would be inside the <data> wrapper element of a reply to the <get> operation. XML attributes SHOULD NOT be present, however if a SW receiving a YANG instance data file encounters XML attributes unknown to it, it MUST ignore them, allowing them to be used later for other purposes.

The JSON format SHALL follow the format of the reply returmed for a RESTCONF GET request directed at the datastore resource: {+restconf}/data. ETags and Timestamps SHOULD NOT be included, but if present SHOULD be ignored.

A YANG Instance data file MUST contain a single instance data set. Instance data MUST conform to the corresponding target YANG Modules and follow the XML/JSON encoding rules as defined in [RFC7950] and [RFC7951] and use UTF-8 character encoding. A single instance data set MAY contain data for any number of target YANG modules, if needed it MAY carry the complete configuraton and state data set for a YANG server. Default values SHOULD NOT but MAY be included. Config=true and config=false data MAY be mixed in the instance data file. Instance data files MAY contain partial data sets. This means mandatory, min-elements or require-instance=true constrains MAY be violated.

The name of the file SHOULD be of the form:

instance-data-set-name ['@' revision-date] ( '.yid' )

E.g. acme-router-modules@2018-01-25.yid

The revision date is optional. It SHOULD NOT be used if the file is stored in a version control system (e.g. git) because the change of file names will break the connection between the different revisions of the file.

Meta data, information about the data set itself SHALL be included in the instance data set. This data will be children of the top level instance-data-set container as defined in the ietf-instance-data YANG module. Meta data SHALL include:

- Name of the instance data set

Meta data SHOULD include:

- Revision date of the instance data set

- Description of the instance data set. The description SHOULD contain information whether and how the data can change during the lifetime of the YANG server.

```
<?xml version="1.0" encoding="UTF-8"?>
<instance-data-set xmlns=
    "urn:ietf:params:xml:ns:yang:ietf-yang-instance-data">
  <name>acme-router-modules</name>
  <revision>2108-01-25</revision>
  <description>Defines the minimal set of modules that any acme-router
       will contain. These modules will always be present.</description>
  <contact>info@acme.com</contact>
  <data>
    <yang-library xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-library">
      <module-set>
        <name>basic</name>
        <module>
          <name>ietf-system</>
          <revision>2014-08-06</revision>
          <!-- description "A later revision may be used."; -->
          <namespace>urn:ietf:params:xml:ns:yang:ietf-system</namespace>
          <feature>authentication</feature>
          <feature>radius-authentication</feature>
        </module>
      </module-set>
    </yang-library>
  </data>
</instance-data-set>
```

Figure 1: XML Instance Data File example

```
{
  "ietf-yang-instance-data:instance-data-set": {
    "name": "acme-router-modules",
    "revision": "2108-01-25",
    "contact": "info@acme.com",
    "description":
      "Defines the set of modules that an acme-router will contain.",
    "data": {
      "ietf-yang-library:yang-library": {
        "module-set": [
          "name": "basic",
          "module": [
            {
              "name": "ietf-system",
              "revision": "2014-08-06",
              "namespace": "urn:ietf:params:xml:ns:yang:ietf-system",
              "feature": ["authentication", "radius-authentication"]
            }
          }
        ]
      }
    }
  }
}
```

Figure 2: JSON Instance Data File example

## 4. YANG Model

```
<CODE BEGINS> file "ietf-yang-instance-data.yang"

module ietf-yang-instance-data {
  yang-version 1.1;
  namespace
    "urn:ietf:params:xml:ns:yang:ietf-yang-instance-data";
  prefix yid ;

  import ietf-yang-data-ext { prefix yd; }

  import ietf-datastores { prefix ds; }

  organization  "IETF NETMOD Working Group";
  contact
   "WG Web:   <https://datatracker.ietf.org/wg/netmodf/>
    WG List:  <mailto:netmod@ietf.org>

    Author:  Balazs Lengyel
       <mailto:balazs.lengyel@ericsson.com>";
```

```
    description "The module defines the structure and content of YANG
         Instance Data Sets.";

    revision 2018-06-30 {
      description  "Initial revision.";
      reference "RFC XXXX: YANG Instance Data";
    }

    yd:yang-data instance-data-format {
      container instance-data-set {
        description "Auxiliary container to carry meta-data for
          the complete instance data set.";

        leaf name {
          type string;
          mandatory true;
          description "Name of a YANG instance data set.";
        }

        leaf description { type string; }

        leaf contact {
          type string;
          description "Contains the same information the contact
              statement carries for a YANG module.";
        }

        leaf organization {
          type string;
          description "Contains the same information the
              organization statement carries for a YANG module.";
        }

        leaf datastore {
          type ds:datastore-ref;
          description  "The identity of the datastore for which
            the instance data is documented for config=true data nodes.
            The leaf MAY be absent in which case the running dtastore or
            if thats not writable, the candidate datastore is implied.

            For config=false data nodes always the operational
            data store is implied.";
        }

        list revision {
          key date;
          description "An instance-data-set SHOULD have at least
              one revision entry.  For every published
```

```
              editorial change, a new one SHOULD be added in front
              of the revisions sequence so that all revisions are
              in reverse chronological order.";

            leaf date {
              type string {
                pattern '\d{4}-\d{2}-\d{2}';
              }
              description "Specifies the data the revision
                  was last modified. Formated as YYYY-MM-DD";
            }

            leaf description { type  string; }
          }

          anydata data {
            mandatory true;
            description "Contains the real instance data.
                The data MUST conform to the relevant YANG Modules.";
          }
        }
      }
    }
```

<CODE ENDS>

## 5. Security Considerations

Depending on the nature of the instance data, instance data files MAY need to be handled in a secure way. The same type of handling should be applied, that would be needed for the result of a <get> operation returning the same data.

## 6. IANA Considerations

To be completed, all the usual requests for a new YANG module

## 7. References

### 7.1. Normative References

[I-D.ietf-netmod-yang-data-ext]
Bierman, A., Bjorklund, M., and K. Watsen, "YANG Data Extensions", draft-ietf-netmod-yang-data-ext-01 (work in progress), March 2018.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

[RFC7951] Lhotka, L., "JSON Encoding of Data Modeled with YANG", RFC 7951, DOI 10.17487/RFC7951, August 2016, <https://www.rfc-editor.org/info/rfc7951>.

## 7.2. Informative References

[I-D.ietf-ccamp-alarm-module]
Vallin, S. and M. Bjorklund, "YANG Alarm Module", draft-ietf-ccamp-alarm-module-01 (work in progress), February 2018.

[I-D.ietf-netconf-rfc7895bis]
Bierman, A., Bjorklund, M., Schoenwaelder, J., Watsen, K., and R. Wilton, "YANG Library", draft-ietf-netconf-rfc7895bis-06 (work in progress), April 2018.

[I-D.ietf-netconf-yang-push]
Clemm, A., Voit, E., Prieto, A., Tripathy, A., Nilsen-Nygaard, E., Bierman, A., and B. Lengyel, "YANG Datastore Subscription", draft-ietf-netconf-yang-push-17 (work in progress), July 2018.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

## Appendix A. Open Issues

- -

## Appendix B. Changes between revisions

v02 - v03

- Added parameter to specify datastore

- Updated the document with the open issues according to the discussions on IETF102

v01 - v02

- The recommendation to document server capabilities was changed to be just the primary use-case. (Merged chapter 4 into the use case chapter.)

- Stated that RFC7950/7951 encoding must be followed which also defines (dis)allowed whitespace rules.

- Added UTF-8 encoding as it is not specified in t950 for instance data

- added XML declaration

v00 - v01

- Redefined using yang-data-ext

- Moved meta data into ordinary leafs/leaf-lists

## Authors' Addresses

Balazs Lengyel
Ericsson
Magyar Tudosok korutja 11
1117 Budapest
Hungary

Phone: +36-70-330-7909
Email: balazs.lengyel@ericsson.com

Benoit Claise
Cisco Systems, Inc.
De Kleetlaan 6a b1
1831 Diegem
Belgium

Phone: +32 2 704 5622
Email: bclaise@cisco.com